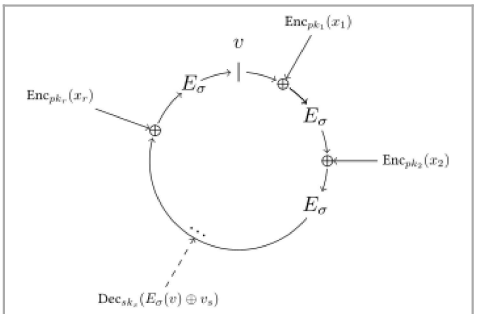
# Homomorphic encryption

**Homomorphic encryption** is a form of encryption that permits users to perform computations on its encrypted data without first decrypting it. These resulting computations are left in an encrypted form which, when decrypted, result in an identical output to that produced had the operations been performed on the unencrypted data. Homomorphic encryption can be used for privacy-preserving outsourced storage and computation. This allows data to be encrypted and out-sourced to commercial cloud environments for processing, all while encrypted.

For sensitive data, such as health care information, homomorphic encryption can be used to enable new services by removing privacy barriers inhibiting data sharing or increase security to existing services. For example, predictive analytics in health care can be hard to apply via a third party service provider due to medical data privacy concerns, but if the predictive analytics service provider can operate on encrypted data instead, these privacy concerns are diminished. Moreover, even if the service provider's system is compromised, the data would remain secure.

**Homomorphic encryption**




| | General |
|---|---|
| **Derived from** | Ring learning with errors |
| **Related to** | Private set intersection<br>Functional encryption |

## Contents



## Description

Homomorphic encryption is a form of encryption with an additional evaluation capability for computing over encrypted data without access to the secret key. The result of such a computation remains encrypted. Homomorphic encryption can be viewed as an extension of public-key cryptography. *Homomorphic* refers to homomorphism in algebra: the encryption and decryption functions can be thought of as homomorphisms between plaintext and ciphertext spaces.

Homomorphic encryption includes multiple types of encryption schemes that can perform different classes of computations over encrypted data.[1] The computations are represented as either Boolean or arithmetic circuits. Some common types of homomorphic encryption are *partially* homomorphic, *somewhat* homomorphic, *leveled fully* homomorphic, and *fully* homomorphic encryption:

- *Partially homomorphic encryption* encompasses schemes that support the evaluation of circuits consisting of only one type of gate, e.g., addition or multiplication.
- *Somewhat homomorphic encryption* schemes can evaluate two types of gates, but only for a subset of circuits.
- *Leveled fully homomorphic encryption* supports the evaluation of arbitrary circuits composed of multiple types of gates of bounded (pre-determined) depth.
- *Fully homomorphic encryption* (FHE) allows the evaluation of arbitrary circuits composed of multiple types of gates of unbounded depth, and is the strongest notion of homomorphic encryption.

For the majority of homomorphic encryption schemes, the multiplicative depth of circuits is the main practical limitation in performing computations over encrypted data. Homomorphic encryption schemes are inherently malleable. In terms of malleability, homomorphic encryption schemes have weaker security properties than non-homomorphic schemes.

# History

Homomorphic encryption schemes have been developed using different approaches. Specifically, fully homomorphic encryption schemes are often grouped into generations corresponding to the underlying approach.[2]

## Pre-FHE

The problem of constructing a fully homomorphic encryption scheme was first proposed in 1978, within a year of publishing of the RSA scheme.[3] For more than 30 years, it was unclear whether a solution existed. During that period, partial results included the following schemes:

- RSA cryptosystem (unbounded number of modular multiplications)
- ElGamal cryptosystem (unbounded number of modular multiplications)
- Goldwasser–Micali cryptosystem (unbounded number of exclusive or operations)
- Benaloh cryptosystem (unbounded number of modular additions)
- Paillier cryptosystem (unbounded number of modular additions)
- Sander-Young-Yung system (after more than 20 years solved the problem for logarithmic depth circuits)[4]
- Boneh–Goh–Nissim cryptosystem (unlimited number of addition operations but at most one multiplication)[5]
- Ishai-Paskin cryptosystem (polynomial-size branching programs)[6]

## First-generation FHE

Craig Gentry, using lattice-based cryptography, described the first plausible construction for a fully homomorphic encryption scheme.[7] Gentry's scheme supports both addition and multiplication operations on ciphertexts, from which it is possible to construct circuits for performing arbitrary computation. The construction starts from a *somewhat homomorphic* encryption scheme, which is limited to evaluating low-degree polynomials over encrypted data; it is limited because each ciphertext is noisy in some sense, and this noise grows as one adds and multiplies ciphertexts, until ultimately the noise makes the resulting ciphertext indecipherable.

Gentry then shows how to slightly modify this scheme to make it *bootstrappable*, i.e., capable of evaluating its own decryption circuit and then at least one more operation. Finally, he shows that any bootstrappable somewhat homomorphic encryption scheme can be converted into a fully homomorphic encryption through a recursive self-embedding. For Gentry's "noisy" scheme, the bootstrapping procedure effectively "refreshes" the ciphertext by applying to it the decryption procedure homomorphically, thereby obtaining a new ciphertext that encrypts the same value as before but has lower noise. By "refreshing" the ciphertext periodically whenever the noise grows too large, it is possible to compute an arbitrary number of additions and multiplications without increasing the noise too much.

Gentry based the security of his scheme on the assumed hardness of two problems: certain worst-case problems over ideal lattices, and the sparse (or low-weight) subset sum problem. Gentry's Ph.D. thesis[8] provides additional details. The Gentry-Halevi implementation of Gentry's original cryptosystem reported timing of about 30 minutes per basic bit operation.[9] Extensive design and implementation work in subsequent years have improved upon these early implementations by many orders of magnitude runtime performance.

In 2010, Marten van Dijk, Craig Gentry, Shai Halevi and Vinod Vaikuntanathan presented a second fully homomorphic encryption scheme,[10] which uses many of the tools of Gentry's construction, but which does not require ideal lattices. Instead, they show that the somewhat homomorphic component of Gentry's ideal lattice-based scheme can be replaced with a very simple somewhat homomorphic scheme that uses integers. The scheme is therefore conceptually simpler than Gentry's ideal lattice scheme, but has similar properties with regards to homomorphic operations and efficiency. The somewhat homomorphic component in the work of Van Dijk et al. is similar to an encryption scheme proposed by Levieil and Naccache in 2008,[11] and also to one that was proposed by Bram Cohen in 1998.[12]

Cohen's method is not even additively homomorphic, however. The Levieil–Naccache scheme supports only additions, but it can be modified to also support a small number of multiplications. Many refinements and optimizations of the scheme of Van Dijk et al. were proposed in a sequence of works by Jean-Sébastien Coron, Tancrède Lepoint, Avradip Mandal, David Naccache, and Mehdi Tibouchi.[13][14][15][16] Some of these works included also implementations of the resulting schemes.

## Second-generation FHE

The homomorphic cryptosystems of this generation are derived from techniques that were developed starting in 2011-2012 by Zvika Brakerski, Craig Gentry, Vinod Vaikuntanathan, and others. These innovations led to the development of much more efficient somewhat and fully homomorphic cryptosystems. These include:

- The Brakerski-Gentry-Vaikuntanathan (BGV, 2011) scheme,[17] building on techniques of Brakerski-Vaikuntanathan;[18]
- The NTRU-based scheme by Lopez-Alt, Tromer, and Vaikuntanathan (LTV, 2012);[19]

- The Brakerski/Fan-Vercauteren (BFV, 2012) scheme,[20] building on Brakerski's *scale-invariant* cryptosystem;[21]
- The NTRU-based scheme by Bos, Lauter, Loftus, and Naehrig (BLLN, 2013),[22] building on LTV and Brakerski's scale-invariant cryptosystem;[21]

The security of most of these schemes is based on the hardness of the (Ring) Learning With Errors (RLWE) problem, except for the LTV and BLLN schemes that rely on an *overstretched*[23] variant of the NTRU computational problem. This NTRU variant was subsequently shown vulnerable to subfield lattice attacks,[24][23] which is why these two schemes are no longer used in practice.

All the second-generation cryptosystems still follow the basic blueprint of Gentry's original construction, namely they first construct a somewhat homomorphic cryptosystem and then convert it to a fully homomorphic cryptosystem using bootstrapping.

A distinguishing characteristic of the second-generation cryptosystems is that they all feature a much slower growth of the noise during the homomorphic computations. Additional optimizations by Craig Gentry, Shai Halevi, and Nigel Smart resulted in cryptosystems with nearly optimal asymptotic complexity: Performing $T$ operations on data encrypted with security parameter $k$ has complexity of only $T \cdot \mathbf{polylog}(k)$.[25][26][27] These optimizations build on the Smart-Vercauteren techniques that enables packing of many plaintext values in a single ciphertext and operating on all these plaintext values in a SIMD fashion.[28] Many of the advances in these second-generation cryptosystems were also ported to the cryptosystem over the integers.[15][16]

Another distinguishing feature of second-generation schemes is that they are efficient enough for many applications even without invoking bootstrapping, instead operating in the leveled FHE mode.

## Third-generation FHE

In 2013, Craig Gentry, Amit Sahai, and Brent Waters (GSW) proposed a new technique for building FHE schemes that avoids an expensive "relinearization" step in homomorphic multiplication.[29] Zvika Brakerski and Vinod Vaikuntanathan observed that for certain types of circuits, the GSW cryptosystem features an even slower growth rate of noise, and hence better efficiency and stronger security.[30] Jacob Alperin-Sheriff and Chris Peikert then described a very efficient bootstrapping technique based on this observation.[31]

These techniques were further improved to develop efficient ring variants of the GSW cryptosystem: FHEW (2014)[32] and TFHE (2016).[33] The FHEW scheme was the first to show that by refreshing the ciphertexts after every single operation, it is possible to reduce the bootstrapping time to a fraction of a second. FHEW introduced a new method to compute Boolean gates on encrypted data that greatly simplifies bootstrapping, and implemented a variant of the bootstrapping procedure.[31] The efficiency of FHEW was further improved by the TFHE scheme, which implements a ring variant of the bootstrapping procedure[34] using a method similar to the one in FHEW.

## Fourth-generation FHE

CKKS scheme[35] supports efficient rounding operations in encrypted state. The rounding operation controls noise increase in encrypted multiplication, which reduces the number of bootstrapping in a circuit. In Crypto2018, CKKS is focused as a solution for encrypted machine learning. (https://www.youtube.com/watch?v=culuNbMPPok&feature=youtu.be&t=3397) This is due to a characteristic of CKKS scheme that encrypts approximate values rather than exact values. When computers store real-valued

data, they remember approximate values with long significant bits, not real values exactly. CKKS scheme is constructed to deal efficiently with the errors arising from the approximations. The scheme is familiar to machine learning which has inherent noises in its structure.

A 2020 article by Baiyu Li and Daniele Micciancio discusses passive attacks against CKKS, suggesting that the standard IND-CPA definition may not be sufficient in scenarios where decryption results are shared.[36] The authors apply the attack to four modern homomorphic encryption libraries (HEAAN, SEAL, HElib and PALISADE) and report that it is possible to recover the secret key from decryption results in several parameter configurations. The authors also propose mitigation strategies for these attacks, and include a Responsible Disclosure in the paper suggesting that the homomorphic encryption libraries already implemented mitigations for the attacks before the article became publicly available. Further information on the mitigation strategies implemented in the homomorphic encryption libraries has also been published.[37][38]

# Partially homomorphic cryptosystems

In the following examples, the notation $\mathcal{E}(x)$ is used to denote the encryption of the message $x$.

### Unpadded RSA

If the RSA public key has modulus $n$ and encryption exponent $e$, then the encryption of a message $m$ is given by $\mathcal{E}(m) = m^e \mod n$. The homomorphic property is then

$$\begin{aligned}\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= m_1^e m_2^e \mod n \\ &= (m_1 m_2)^e \mod n \\ &= \mathcal{E}(m_1 \cdot m_2)\end{aligned}$$

### ElGamal

In the ElGamal cryptosystem, in a cyclic group $G$ of order $q$ with generator $g$, if the public key is $(G, q, g, h)$, where $h = g^x$, and $x$ is the secret key, then the encryption of a message $m$ is $\mathcal{E}(m) = (g^r, m \cdot h^r)$, for some random $r \in \{0, \ldots, q-1\}$. The homomorphic property is then

$$\begin{aligned}\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= (g^{r_1}, m_1 \cdot h^{r_1})(g^{r_2}, m_2 \cdot h^{r_2}) \\ &= (g^{r_1+r_2}, (m_1 \cdot m_2) h^{r_1+r_2}) \\ &= \mathcal{E}(m_1 \cdot m_2).\end{aligned}$$

### Goldwasser–Micali

In the Goldwasser–Micali cryptosystem, if the public key is the modulus $n$ and quadratic non-residue $x$, then the encryption of a bit $b$ is $\mathcal{E}(b) = x^b r^2 \mod n$, for some random $r \in \{0, \ldots, n-1\}$. The homomorphic property is then

$$\begin{aligned}\mathcal{E}(b_1) \cdot \mathcal{E}(b_2) &= x^{b_1} r_1^2 x^{b_2} r_2^2 \mod n \\ &= x^{b_1+b_2} (r_1 r_2)^2 \mod n \\ &= \mathcal{E}(b_1 \oplus b_2).\end{aligned}$$

where $\oplus$ denotes addition modulo 2, (i.e. exclusive-or).

**Benaloh**

In the Benaloh cryptosystem, if the public key is the modulus $n$ and the base $g$ with a blocksize of $c$, then the encryption of a message $m$ is $\mathcal{E}(m) = g^m r^c \mod n$, for some random $r \in \{0, \ldots, n-1\}$. The homomorphic property is then

$$\begin{aligned}\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= (g^{m_1} r_1^c)(g^{m_2} r_2^c) \mod n \\ &= g^{m_1+m_2} (r_1 r_2)^c \mod n \\ &= \mathcal{E}(m_1 + m_2 \mod c).\end{aligned}$$

**Paillier**

In the Paillier cryptosystem, if the public key is the modulus $n$ and the base $g$, then the encryption of a message $m$ is $\mathcal{E}(m) = g^m r^n \mod n^2$, for some random $r \in \{0, \ldots, n-1\}$. The homomorphic property is then

$$\begin{aligned}\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= (g^{m_1} r_1^n)(g^{m_2} r_2^n) \mod n^2 \\ &= g^{m_1+m_2} (r_1 r_2)^n \mod n^2 \\ &= \mathcal{E}(m_1 + m_2).\end{aligned}$$

**Other partially homomorphic cryptosystems**

- Okamoto–Uchiyama cryptosystem
- Naccache–Stern cryptosystem
- Damgård–Jurik cryptosystem
- Sander–Young–Yung encryption scheme
- Boneh–Goh–Nissim cryptosystem
- Ishai–Paskin cryptosystem
- Castagnos–Laguillaumie cryptosystem[39]

# Fully homomorphic encryption

A cryptosystem that supports *arbitrary computation* on ciphertexts is known as fully homomorphic encryption (FHE). Such a scheme enables the construction of programs for any desirable functionality, which can be run on encrypted inputs to produce an encryption of the result. Since such a program need never decrypt its inputs, it can be run by an untrusted party without revealing its inputs and internal state. Fully homomorphic cryptosystems have great practical implications in the outsourcing of private computations, for instance, in the context of cloud computing.[40]

## Implementations

A list of open-source FHE libraries implementing second-generation and/or third-generation FHE schemes is provided below. An up-to-date list of homomorphic encryption implementations (https://github.com/jonaschn/awesome-he) is also maintained by the community on GitHub.

There are several open-source implementations of second- and third-generation fully homomorphic encryption schemes. Second-generation FHE scheme implementations typically operate in the leveled FHE mode (though bootstrapping is still available in some libraries) and support efficient SIMD-like packing of data; they are typically used to compute on encrypted integers or real/complex numbers. Third-generation FHE scheme implementations often bootstrap after each Boolean gate operation but have limited support for packing and efficient arithmetic computations; they are typically used to compute Boolean circuits over encrypted bits. The choice of using a second-generation vs. third-generation scheme depends on the input data types and the desired computation.

FHE libraries

| Name | Developer | BGV[17] | CKKS[35] | BFV[20] | FHEW[32] | CKKS Bootstrapping[41] | TFHE[33] | Description |
|---|---|---|---|---|---|---|---|---|
| HElib[42] | IBM | Yes | Yes | No | No | No | No | BGV scheme with the GHS optimizations. |
| Microsoft SEAL[43] | Microsoft | No | Yes | Yes | No | No | No | |
| PALISADE[44] | Consortium of DARPA-funded defense contractors and academics: New Jersey Institute of Technology, Duality Technologies, Raytheon BBN Technologies, MIT, University of California, San Diego and others. | Yes | Yes | Yes | Yes | No | Yes | General-purpose lattice cryptography library. |
| HEAAN[45] | Seoul National University | No | Yes | No | No | Yes | No | |
| FHEW[32] | Leo Ducas and Daniele Micciancio | No | No | No | Yes | No | No | |
| TFHE[33] | Ilaria Chillotti, Nicolas Gama, Mariya Georgieva and Malika Izabachene | No | No | No | No | No | Yes | |
| FV-NFLlib[46] | CryptoExperts | No | No | Yes | No | No | No | |
| NuFHE[47] | NuCypher | No | No | No | No | No | Yes | Provides a GPU implementation of TFHE. |
| Lattigo[48] | EPFL-LDS (https://lds.epfl.ch/) | No | Yes | Yes | No | Yes[49] | No | Implementation in Go along with their distributed variants[50] enabling Secure multi-party computation. |

FHE frameworks

| Name | Developer | FHEW [32] | TFHE | Helib | SEAL | PALISADE |
|---|---|---|---|---|---|---|
| E3[51] | MoMA Lab at NYU Abu Dhabi | Yes | Yes | Yes | Yes | Yes |
| SHEEP[52] | Alan Turing Institute | No | Yes | Yes | Yes | Yes |

### Standardization

A community standard for homomorphic encryption is maintained by the HomomorphicEncryption.org (http://homomorphicencryption.org/) group, an open industry/government/academia consortium co-founded in 2017 by Microsoft, IBM and Duality Technologies. The current standard document (http://homomorphicencryption.org/standard) includes specifications of secure parameters for RLWE.

## See also

- Homomorphic secret sharing
- Homomorphic signatures for network coding
- Private biometrics
- Verifiable computing using a fully homomorphic scheme
- Client-side encryption
- Searchable symmetric encryption
- Secure multi-party computation
- Format-preserving encryption
- Polymorphic code
- Private set intersection

## References


1. Armknecht, Frederik; Boyd, Colin; Gjøsteen, Kristian; Jäschke, Angela; Reuter, Christian; Strand, Martin (2015). "A Guide to Fully Homomorphic Encryption" (https://eprint.iacr.org/2015/1192).
2. Vinod Vaikuntanathan. "Homomorphic Encryption References" (https://people.csail.mit.edu/vinodv/FHE/FHE-refs.html).
3. R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, 1978.
4. Sander, Tomas; Young, Adam L.; Yung, Moti (1999). *Non-Interactive CryptoComputing For NC1*. *Focs1991*. pp. 554–566. doi:10.1109/SFFCS.1999.814630 (https://doi.org/10.1109%2FSFFCS.1999.814630). ISBN 978-0-7695-0409-4. S2CID 1976588 (https://api.semanticscholar.org/CorpusID:1976588).
5. D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In *Theory of Cryptography Conference*, 2005.
6. Y. Ishai and A. Paskin. Evaluating branching programs on encrypted data. In *Theory of Cryptography Conference*, 2007.
7. Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices (http://portal.acm.org/citation.cfm?id=1536414.1536440). In *the 41st ACM Symposium on Theory of Computing (STOC)*, 2009.
8. Craig Gentry. "A Fully Homomorphic Encryption Scheme (Ph.D. thesis)" (http://crypto.stanford.edu/craig/) (PDF).
9. Gentry, Craig; Halevi, Shai (2010). "Implementing Gentry's fully-homomorphic encryption scheme" (http://eprint.iacr.org/2010/520). *Eurocrypt 2011*.

10. Van Dijk, Marten; Gentry, Craig; Halevi, Shai; Vinod, Vaikuntanathan (2009). "Fully Homomorphic Encryption over the Integers" (http://eprint.iacr.org/2009/616). *Eurocrypt 2010*.
11. Levieil, Eric; Naccache, David. "Cryptographic Test Correction" (https://www.iacr.org/archive/pkc2008/49390088/49390088.pdf) (PDF).
12. Cohen, Bram. "Simple Public Key Encryption" (https://web.archive.org/web/20111007060226/http://bramcohen.com/simple_public_key.html). Archived from the original (http://bramcohen.com/simple_public_key.html) on 2011-10-07.
13. Coron, Jean-Sébastien; Naccache, David; Tibouchi, Mehdi (2011). "Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers" (http://eprint.iacr.org/2011/440). *Eurocrypt 2012*.
14. Coron, Jean-Sébastien; Mandal, Avradip; Naccache, David; Tibouchi, Mehdi (2011). "Fully Homomorphic Encryption over the Integers with Shorter Public Keys" (http://eprint.iacr.org/2011/441). *Crypto 2011*. Lecture Notes in Computer Science. **6841**: 487–504. doi:10.1007/978-3-642-22792-9_28 (https://doi.org/10.1007%2F978-3-642-22792-9_28). ISBN 978-3-642-22791-2.
15. Coron, Jean-Sébastien; Lepoint, Tancrède; Tibouchi, Mehdi (2013). "Batch Fully Homomorphic Encryption over the Integers" (http://eprint.iacr.org/2013/036). *Eurocrypt 2013*.
16. Coron, Jean-Sébastien; Lepoint, Tancrède; Tibouchi, Mehdi (2014). "Scale-Invariant Fully Homomorphic Encryption over the Integers" (http://eprint.iacr.org/2014/032). *PKC 2014*.
17. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully Homomorphic Encryption without Bootstrapping (http://eprint.iacr.org/2011/277), In *ITCS 2012*
18. Z. Brakerski and V. Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE (http://eprint.iacr.org/2011/344). In *FOCS 2011* (IEEE)
19. A. Lopez-Alt, E. Tromer, and V. Vaikuntanathan. On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption (https://eprint.iacr.org/2013/094). In *STOC 2012* (ACM)
20. Fan, Junfeng; Vercauteren, Frederik (2012). "Somewhat Practical Fully Homomorphic Encryption" (https://eprint.iacr.org/2012/144).
21. Z. Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP (http://eprint.iacr.org/2012/078), In *CRYPTO 2012* (Springer)
22. J. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme (https://eprint.iacr.org/2013/075). In *IMACC 2013* (Springer)
23. M. Albrecht, S. Bai, and L. Ducas. A subfield lattice attack on overstretched NTRU assumptions (https://eprint.iacr.org/2016/127), In *CRYPTO 2016* (Springer)
24. Cheon, J. H.; Jeong, J; Lee, C. (2016). "An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low-level encoding of zero" (https://doi.org/10.1112%2FS1461157016000371). *LMS Journal of Computation and Mathematics*. **19** (1): 255–266. doi:10.1112/S1461157016000371 (https://doi.org/10.1112%2FS1461157016000371).
25. C. Gentry, S. Halevi, and N. P. Smart. Fully Homomorphic Encryption with Polylog Overhead (http://eprint.iacr.org/2011/566). In *EUROCRYPT 2012* (Springer)
26. C. Gentry, S. Halevi, and N. P. Smart. Better Bootstrapping in Fully Homomorphic Encryption (http://eprint.iacr.org/2011/680). In *PKC 2012* (SpringeR)
27. C. Gentry, S. Halevi, and N. P. Smart. Homomorphic Evaluation of the AES Circuit (http://eprint.iacr.org/2012/099). In *CRYPTO 2012* (Springer)
28. Smart, Nigel P.; Vercauteren, Frederik (2014). "Fully Homomorphic SIMD Operations" (http://eprint.iacr.org/2011/133). *Designs, Codes and Cryptography*. **71** (1): 57–81. doi:10.1007/s10623-012-9720-4 (https://doi.org/10.1007%2Fs10623-012-9720-4). S2CID 11202438 (https://api.semanticscholar.org/CorpusID:11202438).
29. C. Gentry, A. Sahai, and B. Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based (http://eprint.iacr.org/2013/340). In *CRYPTO 2013* (Springer)

30. Z. Brakerski and V. Vaikuntanathan. Lattice-Based FHE as Secure as PKE (http://eprint.iacr.org/2013/541). In *ITCS 2014*
31. J. Alperin-Sheriff and C. Peikert. Faster Bootstrapping with Polynomial Error (http://eprint.iacr.org/2014/094). In *CRYPTO 2014* (Springer)
32. Leo Ducas; Daniele Micciancio. "FHEW: A Fully Homomorphic Encryption library" (https://github.com/lducas/FHEW). Retrieved 31 December 2014.
33. Ilaria Chillotti; Nicolas Gama; Mariya Georgieva; Malika Izabachene. "Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 Seconds" (https://tfhe.github.io/tfhe). Retrieved 31 December 2016.
34. N. Gama, M. Izabachène, P.Q. Nguyen, and X. Xie Structural Lattice Reduction: Generalized Worst-Case to Average-Case Reductions and Homomorphic Cryptosystems (https://eprint.iacr.org/2014/283). In *EUROCRYPT 2016* (Springer)
35. Cheon, Jung Hee; Kim, Andrey; Kim, Miran; Song, Yongsoo (2017). "Homomorphic encryption for arithmetic of approximate numbers". *Takagi T., Peyrin T. (eds) Advances in Cryptology – ASIACRYPT 2017*. ASIACRYPT 2017. Springer, Cham. pp. 409–437. doi:10.1007/978-3-319-70694-8_15 (https://doi.org/10.1007%2F978-3-319-70694-8_15).
36. Li, Baily; Micciancio, Daniele (2020). "On the Security of Homomorphic Encryption on Approximate Numbers" (https://eprint.iacr.org/2020/1533.pdf) (PDF). *IACR ePrint Archive 2020/1533*.
37. Cheon, Jung Hee; Hong, Seungwan; Kim, Duhyeong (2020). "Remark on the Security of CKKS Scheme in Practice" (https://eprint.iacr.org/2020/1581.pdf) (PDF). *IACR ePrint Archive 2020/1581*.
38. "Security of CKKS" (https://palisade-crypto.org/security-of-ckks). Retrieved 10 March 2021.
39. Guilhem Castagnos and Fabien Laguillaumie (2015). "Linearly Homomorphic Encryption from DDH" (https://eprint.iacr.org/2015/047.pdf) (PDF).
40. Daniele Micciancio (2010-03-01). "A First Glimpse of Cryptography's Holy Grail" (http://cacm.acm.org/magazines/2010/3/76275-a-first-glimpse-of-cryptographys-holy-grail/fulltext). Association for Computing Machinery. p. 96. Retrieved 2010-03-17.
41. Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim and Yongsoo Song. Bootstrapping for Approximate Homomorphic Encryption (https://eprint.iacr.org/2018/153). In *EUROCRYPT 2018 (Springer)*.
42. Shai Halevi; Victor Shoup. "HElib: An Implementation of homomorphic encryption" (https://github.com/homenc/HElib). Retrieved 31 December 2014.
43. Microsoft Research. "Microsoft SEAL" (https://www.microsoft.com/en-us/research/project/microsoft-seal). Retrieved 20 February 2019.
44. "PALISADE Lattice Cryptography Library" (http://palisade-crypto.org). Retrieved 1 January 2019.
45. Jung Hee Cheon; Kyoohyung Han; Andrey Kim; Miran Kim; Yongsoo Song. "Homomorphic Encryption for Arithmetic of Approximate Numbers" (https://github.com/snucrypto/HEAAN). Retrieved 15 May 2016.
46. Crypto Experts. "FV-NFLlib" (https://github.com/CryptoExperts/FV-NFLlib). Retrieved 1 November 2019.
47. NuCypher. "A GPU implementation of fully homomorphic encryption on torus" (https://github.com/nucypher/nufhe). Retrieved 1 November 2019.
48. EPFL-LDS. "Lattigo v2.1.1" (https://github.com/ldsec/lattigo). Retrieved 15 March 2021.
49. Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza and Jean-Pierre Hubaux. Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-Sparse Keys (https://eprint.iacr.org/2020/1203). In *EUROCRYPT 2021 (Springer)*.
50. Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat and Jean-Pierre Hubaux. Multiparty Homomorphic Encryption from Ring-Learning-With-Errors (https://eprint.iacr.org/2020/304).
51. MoMA Lab, New York University Abu Dhabi (2019-07-24). "Encrypt-Everything-Everywhere (E3)" (https://github.com/momalab/e3). Retrieved 27 July 2019.

52. Alan Turing Institute, London, UK (2019-11-01). "SHEEP, a Homomorphic Encryption Evaluation Platform" (https://github.com/alan-turing-institute/SHEEP). Retrieved 1 November 2019.

# External links

- Daniele Micciancio's FHE references (http://cseweb.ucsd.edu/~daniele/LatticeLinks/FHE.html)
- Vinod Vaikuntanathan's FHE references (https://people.csail.mit.edu/vinodv/FHE/FHE-refs.html)
- "Alice and Bob in Cipherspace" (https://www.americanscientist.org/article/alice-and-bob-in-cipherspace). *American Scientist*. September 2012. Retrieved 2018-05-08.
- HElib (https://github.com/shaih/HElib), Microsoft SEAL (https://www.microsoft.com/en-us/research/project/microsoft-seal), PALISADE (http://palisade-crypto.org), HEAAN (https://github.com/snucrypto/HEAAN), FHEW (https://github.com/lducas/FHEW), and TFHE (https://tfhe.github.io/tfhe) (open-source homomorphic encryption libraries)